Chapter 2
Error Tolerant Predictive Control Based on Recurrent Neural Models

Petia Georgieva and Sebastião Feyo de Azevedo

Abstract This chapter is focused on developing a feasible model predictive control (MPC) based on time dependent recurrent neural network (NN) models. A modification of the classical regression neural models is proposed suitable for prediction purposes. In order to reduce the computational complexity and to improve the prediction ability of the neural model, optimization of the NN structure (lag space selection, number of hidden nodes), pruning techniques and identification strategies are discussed. Furthermore a computationally efficient modification of the general nonlinear MPC is proposed termed Error Tolerant MPC (ETMPC). The NN model is imbedded into the structure of the ETMPC and extensively tested on a dynamic simulator of an industrial crystalizer. The results demonstrate that the NN-based ETMPC relaxes the computational burden without losing closed loop performance and can complement other solutions for feasible industrial real time control.

2.1 Introduction

The concept of Model-based Predictive Control (MPC) was introduced in the eighties [1]. It does not refer to a particular control method, instead it corresponds to a general control framework [2]. MPC is known to be the most successful advanced control approach in practical applications, representing a true alternative to the classical Proportional Integral Derivative (PID) control. The main reasons

P. Georgieva (✉)
Signal Processing Lab, Department of Electronics Telecommunications and Informatics (DETI), Institute of Electronics Engineering and Telematics of Aveiro (IEETA), University of Aveiro, Aveiro, Portugal
e-mail: petia@ua.pt

S. F. de Azevedo
Faculdade de Engenharia de Universidade do Porto, Porto, Portugal

V. E. Balas et al. (eds.), *Innovations in Intelligent Machines-5*,
Studies in Computational Intelligence 561, DOI: 10.1007/978-3-662-43370-6_2,

for this are its potential i) to take directly into account the process input/output constrains, ii) to consider multiple process objectives and iii) to control processes with nonlinear time varying dynamics. The main difference between the existing MPC configurations is in the model used to predict the future behavior of the process or the implemented optimization procedure. The first industrial applications of MPC were based on linear models [3], later on successful applications of nonlinear MPC (NMPC) [4] were also reported. Despite the recognized progress of NMPC, its on-line implementation with predictions running on a large number of nonlinear differential and algebraic equations (DAE), i.e. the first principles process model, is a huge challenge. Such predictions may lead to feasibility problems for processes with fast nonlinear dynamics (as for example chemical crystallization/precipitation processes) [5] or can stuck into numerical problems as stiffness or ill-conditioning. Moreover, in many cases development of first principles models based on physical laws is difficult and time consuming [6].

Several suggestions have been made to deal with these problems ranging from simple extension of Dynamic Matrix Control (DMC) based on successive linearization of the nonlinear model to more elaborate techniques involving discretization of the model followed by solution via non-linear programming [7]. These solutions are usually computationally very intensive, assuming unlimited computing resources, which is only valid for high value products and industries with state of the art control equipment.

MPC algorithms based on artificial Neural Network (NN) models are a promising alternative that addresses also the above problems. Among various NN structures, one-step ahead predictors where the neural model is trained non-recurrently is the most typical option e.g., [8, 9]. However, if the neural models are used for long range prediction (usually the case with MPC), the prediction error will be propagated. Another option is to use a multi-model [10, 11]. For each sampling instant within the prediction horizon an independent submodel is used, thus the prediction error is not propagated. A third option is to use specialised recurrent training algorithms for neural models [7, 12, 13], but they are significantly more computationally demanding in comparison with one-step ahead predictor training, and the obtained models may be sensitive to noise.

The contribution of this chapter is twofold. First, it introduces a recurrent multistep ahead predictive neural model where the past model predictions are substituted by the process measurements and thus the prediction error is not propagated. Secondly, in order to further improve the efficiency and reduce the complexity of the control system a modification of the general NMPC is also proposed, where the NMPC is executed only when the tracking error is outside a pre-specified bound. Once the error converges towards the tolerance zone, the NMPC is switched off and the control action is kept constant. The combination of the proposed neural model and the introduced error tolerance (ET) in the optimization represents a promising compromise between process performance and computational complexity and can complement other suggestions in the literature for feasible industrial real time control. This modification is termed NN-based Error Tolerant Model Predictive Controller (NN ETMPC).

2.2 Model Predictive Control Algorithms

Although individually different in form, the underlying idea of all MPC schemes can be summarized as follows [14]: i) A dynamic model and on-line measurements are used to predict the future process behavior; ii) On the basis of the process output predictions over a prediction horizon (H_p), optimization is performed to find a sequence of manipulated input moves over a control horizon (H_c) that minimizes a chosen cost function while satisfying all given constrains; iii) Only the first of the calculated input sequence is implemented and the whole optimization is repeated at the next sampling time (see Fig. 2.1).

The on-line NMPC implementation has often a discrete form, where the exact optimization is substituted by a discrete approximation [15]. The continuous time $t \in [t_0, t_f]$ is divided into equally spaced time intervals, with discrete time steps $t_k = t_0 + k\Delta t$ and $k = 0, 1, \ldots, N$. At each consecutive sampling instant (k), a set of future ($k + p$) control increments $\Delta u(k + p/k) = u(k + p/k) - u(k + p - 1/k)$ is calculated

$$\Delta \mathbf{u}(k) = [\Delta u(k/k), \Delta u(k+1/k), \ldots \Delta u(k + H_c - 1/k)] \tag{2.1}$$

The following quadratic cost function is typically used:

$$\min_{\Delta u(k/k), \Delta u(k+1/k) \ldots\ldots \Delta u(k+H_c-1/k)} J = \lambda_1 \sum_{p=1}^{H_p} (e(k+p/k))^2 + \lambda_2 \sum_{p=1}^{H_c} (\Delta u(k+p/k))^2 \tag{2.2}$$

where $e(k+p/k) = ref(k+p/k) - \hat{y}(k+p/k)$. Note that the first term in (2.2) is related with the objective to minimize the deviation of the predicted values of the output $\hat{y}(k+p/k)$ from the respective reference $ref\ (k + p/k)$ over the prediction horizon. The second term penalizes excessive control increments. The prediction horizon (H_p) is the number of future time steps over which the prediction errors are minimized and the control horizon (H_c) is the number of future time steps over which the control increments are minimized, $H_p \geq H_c$. It is assumed that the control increments for sampling instants after the control horizon are zero ($\Delta u(k + p/k) = 0, p \geq H_c$). Parameters $\{\lambda_1, \lambda_2\} \in \Re \geq 0$ determine the contribution of each term of (2.2) and consider also the problem of different numerical ranges.

The cost function (2.2) has to be minimized subject to the following constraints:

$$u_{\min} \leq u(k+p/k) \leq u_{\max}, \quad p = 0, 1, \ldots H_c - 1 \tag{2.3}$$

$$\Delta u_{\min} \leq \Delta u(k+p/k) \leq \Delta u_{\max}, \quad p = 0, 1, \ldots H_c - 1 \tag{2.4}$$

$$y_{\min} \leq \hat{y}(k+p/k) \leq y_{\max}, \quad p = 1, 2, \ldots H_p \tag{2.5}$$

$u_{\min}$ and $u_{\max}$ are the lower and the upper bounds of the manipulated inputs, while $\Delta u_{\min}$ and $\Delta u_{\max}$ are the limits of the manipulated input increments.

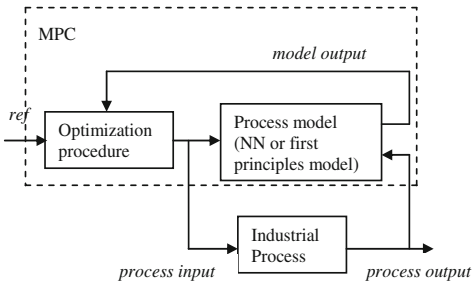

Fig. 2.1 Model based predictive control (MPC) scheme

Constraints (2.3–2.4) are usually related with actuator saturation or rate-of-change restrictions, whereas constraints (2.5) are associated with operational limits ($y_{\min}$, $y_{\max}$) such as equipment specifications and safety considerations. Only the first element of the determined sequence (2.1) is actually applied to the process

$$u(k) = \Delta u(k/k) + u(k-1) \tag{2.6}$$

The general prediction equation for $p = 1, 2, \ldots H_p$ is

$$\hat{y}(k+p/k) = y_m(k+p/k) + dist(k) \tag{2.7}$$

where quantities $y_m(k + p/k)$ are calculated from a dynamical model of the process. The unmeasured disturbance $dist(k)$ is estimated from

$$dist(k) = y(k) - y_m(k/k-1) \tag{2.8}$$

where $y(k)$ is measured from the process and $y_m(k/k - 1)$is calculated from the dynamical model. $dist(k)$ is assumed to be constant over the prediction horizon $dist(k + p/k) = dist(k)$, $i = 1, \ldots H_c$.

MPC algorithms use an explicit dynamical model in order to predict the future process behavior (Fig. 2.1). Therefore, the main issue to address is the choice of the process model structure since it affects the performance and accuracy of the control action. Models based on physical laws (first principles models) are usually very precise, however not suitable for on-line control since they are complicated and may lead to numerical problems. Linear models are approximations and a good choice when the process nonlinearity is mild. However, when the process is substantially nonlinear, data-based models as fuzzy and neural network structures are gaining more popularity [10].

On Fig. 2.2 are summarized the most typical MPC algorithms. When the model is linear and there are no constrains affecting the process, an analytical optimal solution can be found in a closed form [2]. This is however an idealization, while in practice the linear MPC is solved numerically at each sampling instant, by the constrained quadratic programming (QP) optimization. When the model is nonlinear, then the optimization problem (2.2) is certainly not linear quadratic. It is generally a nonconvex and even multimodal one. For such problems there are no

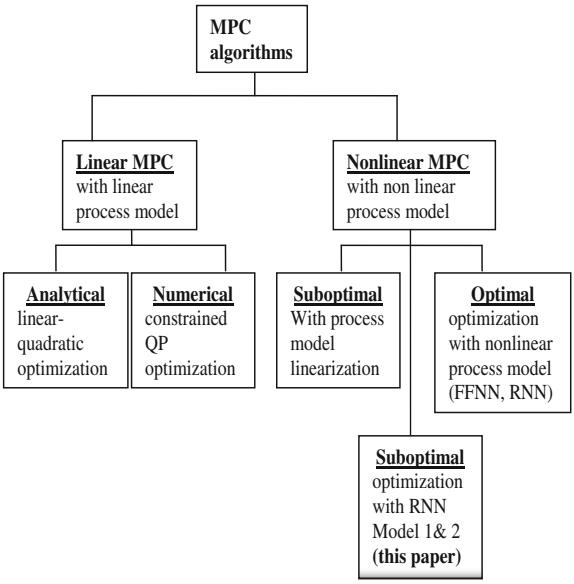

Fig. 2.2 MPC algorithms

sufficiently fast and reliable numerical optimization procedures, i.e., procedures yielding always an optimal point and within predefined time limit—as is required in on-line control applications [13]. Therefore, many attempts have been made to construct simplified (and generally suboptimal) nonlinear MPC algorithms avoiding full on-line nonlinear optimization, first of all using model linearizations or multiple linear models (for example fuzzy structures) [11]. On the other hand, successful optimal MPC based on nonlinear optimization (MPC-NO) are mainly those applying NN techniques. However, the MPC-NO algorithm with NN is still computationally very demanding (nonconvex, local minima) and significantly more powerful controllers are required especially for shorter sampling periods.

How to reduce the computational complexity of the nonlinear MPC is therefore a pertinent problem. The contribution of this chapter is the combination of a relaxed cost function (suboptimal optimization) and the specific time dependent Recurrent Neural Network (RNN) structure in order to reduce the MPC computational burden.

2.3 Neural Dynamical Process Model

NNs became an established methodology for identification and control of nonlinear systems because they are universal approximators, have a relatively small number of parameters and a simple structure. The NN-based control can be approached in direct or indirect control framework. Direct neural control means

that the controller itself is a NN, while in the indirect neural control scheme, first a NN is used to model the process to be controlled, and this model is then embedded in the control structure [16]. The first approach consists of a number of methods such as NN Inverse Control (known also as Adaptive Neural Control), NN Internal Model Control (IMC), NN feedback linearization, NN feedforward controller + conventional feedback controller [17]. The second approach is associated with Direct adaptive control and NN-based MPC [18].

The main MPC approaches based on NN dynamical models can be briefly summarized as:

(1) MPC in which the neural model is used directly, without any simplifications. The control action is computed by a nonlinear optimisation routine, e.g., [15, 16].
(2) MPC in which the neural model is linearised on-line. At each sampling instant the control action is computed by a QP routine, e.g., [10, 13].
(3) Approximate neural MPC in which the NN replaces the whole control algorithm. The network generates the control actions [17].
(4) There are some specific versions of the above approaches termed stable neural MPC [19], adaptive neural MPC [20], robust neural MPC [21].

All these algorithms use a one step ahead predictive structure for NN training, while the MPC implementation of the trained neural model is as a multi step ahead process predictor. This discrepancy may cause biased predictions and error accumulation. We propose here to use the same structure for training and MPC implementation by making changes of the classical regression neural structure.

The most typical NN regression models are inspired by the classical nonlinear function approximation techniques [22].

2.3.1 NN Final Impulse Response (NNFIR) Model

$$y_m(k) = g[u(k-\tau), \ldots u(k-\tau-n_u+1)] \tag{2.9}$$

At each discrete moment k, the model output, $y_m(k)$, is a function only of n_u past process inputs $u(k-\tau)\ldots. u(k-\tau-n_u+1)$, τ is the discrete time delay, $\tau \leq n_u$.

2.3.2 NN Auto Regressive with eXogenous Input (NNARX) Model

$$y_m(k) = g\left\lfloor y(k-1), \ldots y(k-n_y), u(k-\tau), \ldots u(k-\tau-n_u+1)\right\rfloor \tag{2.10}$$

The network input consists of n_u past process (exogenous) inputs and n_y past process outputs $y(k-1)\ldots. y(k-n_y)$.

2.3.3 NN Auto Regressive Moving Average with eXogenous Input (NNARMAX) Model

$$y_m(k) = g\left[y(k-1), \ldots y(k-n_y), u(k-\tau), \ldots u(k-\tau-n_u+1), e(k-1), \ldots e(k-n_e)\right] \tag{2.11}$$

where $e(k) = y(k) - y_m(k)$.

The model output $y_m(k)$is a function of n_u past process inputs, n_y past process outputs and n_e past errors between the process and the model output $e(k-1)\ldots e(k-n_e)$.

2.3.4 NN Output Error (NNOE) Model

$$y_m(k) = g\left[y_m(k-1), \ldots y_m(k-n_y), u(k-\tau), \ldots u(k-\tau-n_u+1)\right] \tag{2.12}$$

The network input consists of n_u past process inputs and n_y past model outputs $y_m(k-1)\ldots y_m(k-n_y)$.

NNFIR or NNARX models can be designed as a Feed Forward NN (FFNN), while to build a NNARMAX or NNOE model, a Recurrent Neural Network (RNN) with global feedback (network outputs are fed back as network inputs) is required. RNNs are more adequate to approximate process dynamics since they can encode the system real time memory and have usually a shorter lag space (n_y and n_u) [23].

The output of any of the four NN models (A–D) can be computed as

$$y_m(k) = a_0 + \sum_{j=1}^{H} a_j \varphi\left(z_j(k)\right) \tag{2.13}$$

where H is the number of hidden nodes and φ is the nonlinear activation function. $z_j(k)$ is the sum of inputs of the ith hidden node. For a standard RNN

$$z_j(k) = b_{j0} + \sum_{i=1}^{n_y} b_{ji} y(k-i) + \sum_{i=1}^{n_u} c_{ji} u(k-\tau-i+1) \tag{2.14}$$

$\{a_j, b_{j\text{-}}, c_{j\text{-}}\} \in \Re$ are the hidden-to-output layer weights and the input-to-hidden layer weights, respectively.

MPC requires multistep ahead predictions of the process outputs. If NNARX or NNARMAX are used as predictive models, according to (2.10) and (2.11), future process outputs (measurements) are necessary. For example, let $n_y = n_u = 3$,

$\tau = 1$, $H_p = 4$, $H_c = 2$. Assuming NNARX model structure, at each sampling moment k the following computations will be performed

$$\begin{aligned} y_m(k) &= g[y(k-1), y(k-2), y(k-3), u(k-1), u(k-2), u(k-3)] \\ y_m(k+1) &= g[y(k), y(k-1), y(k-2), u(k), u(k-1), u(k-2)] \\ y_m(k+2) &= g[y(k+1), y(k), y(k-1), u(k+1), u(k), u(k-1)] \\ y_m(k+3) &= g[y(k+2), y(k+1), y(k), u(k+2), u(k+1), u(k)] \\ y_m(k+4) &= g[y(k+3), y(k+2), y(k+1), u(k+2, u(k+2), u(k+1)] \end{aligned} \tag{2.15}$$

In order to compute $y_m(k+2)$, $y_m(k+3)$, $y_m(k+3)$ future process outputs and future values of the control signal (i.e. the decision variables of the MPC algorithm) are required. For NNARMAX model, future errors $[y(k+p) - y_m(k+p)]$ are additionally required. Since future process measurements are not available, the NNARX or NNARMAX can be used only for once step ahead prediction where the neural model output is a function of the current and past input-output process data.

The NNOE model (2.12) seems more adequate for multi-step ahead predictions because it is a completely recurrent structure depending only on the process inputs and the fed back model predictions:

$$\begin{aligned} y_m(k) &= g[y_m(k-1), y_m(k-2), y_m(k-3), u(k-1), u(k-2), u(k-3)] \\ y_m(k+1) &= g[y_m(k), y_m(k-1), y_m(k-2), u(k), u(k-1), u(k-2)] \\ y_m(k+2) &= g[y_m(k+1), y_m(k), y_m(k-1), u(k+1), u(k), u(k-1)] \\ y_m(k+3) &= g[y_m(k+2), y_m(k+1), y_m(k), u(k+2), u(k+1), u(k)] \\ y_m(k+4) &= g[y_m(k+3), y_m(k+2), y_m(k+1), u(k+2, u(k+2), u(k+1)] \end{aligned} \tag{2.16}$$

According to (2.16) the output of the NNOE model does not use process output measurements. However, in case of model inaccuracies and under parameterization, prediction errors can be accumulated and jeopardize the control system.

We propose here a predictive neural model that is a mixture of NNARX and NNOE and substitute the past model outputs with past process measurements. In the example above $y_m(k-1)$, $y_m(k-2)$, $y_m(k-3)$ in (2.16) will be substituted by $y(k-1)$, $y(k-2)$, $y(k-3)$. Then, the predictive form of (2.13) is

$$y_m(k+p/k) = a_0 + \sum_{j=1}^{H} a_j \, \varphi\big(z_j(k+p/k)\big) \tag{2.17}$$

The proposed modification (Model 1) is comparatively studied with the NNOE structure (Model 2).

2.3.4.1 On-line Multistep Ahead Predictive Model1

$$
\begin{aligned}
z_j(k+p/k) = b_{j0} &+ \sum_{i=1}^{\text{var 1}} b1_{ji} y_m(k-i+p/k) \rightarrow \textbf{future model predictions} \\
&+ \sum_{i=1}^{n_y-\text{var1}} b2_{ji} y(k-i) \rightarrow (\textbf{measured}(\textbf{past})\textbf{processoutputs}) \\
&+ \sum_{i=1}^{\text{var 2}} c1_{ji} u(k-\tau+1-i+p/k) \rightarrow (\text{future input signals}) \\
&+ \sum_{i=1}^{n_u-\text{var 2}} c2_{ji} u(k-\tau+1-i) \rightarrow (\text{past input signals})
\end{aligned}
\tag{2.18}
$$

2.3.4.2 On-line Multistep Ahead Predictive Model 2 (NNOE)

$$
\begin{aligned}
z_j(k+p/k) = b_{j0} &+ \sum_{i=1}^{\text{var 1}} b1_{ji} y_m(k-i+p/k) \rightarrow \textbf{future model predictions} \\
&+ \sum_{i=1}^{n_y-\text{var 1}} b2_{ji} y(k-i) \rightarrow (\textbf{past model outputs}) \\
&+ \sum_{i=1}^{\text{var 2}} c1_{ji} u(k-\tau+1-i+p/k) \rightarrow (\text{future input signals}) \\
&+ \sum_{i=1}^{n_u-\text{var 2}} c2_{ji} u(k-\tau+1-i) \rightarrow (\text{past input signals})
\end{aligned}
\tag{2.19}
$$

2.3.4.3 Neural Model Structure for Training and MPC Implementation

The training stage of any of the regression models (A–D) is usually based on the NN structure (2.13)–(2.14) where the objective is to minimize the Sum of the Squared Errors (SSE)

$$SSE = \sum_{k \in data\ set} [d(k) - y_m(k/k-1)]^2 \tag{2.20}$$

$y_m(k/k-1)$ denotes the output of the neural model for the current sampling instant k calculated using signals up to the sampling instant $k-1$ and $d(k)$ is the process output collected during the identification experiment. Such training ignores the multi step ahead predictive role of the model (2.17) in the MPC framework. Due to the different neural model structures used for training and MPC implementation, prediction errors are further propagated.

In order to overcome this problem, during the training of Models 1 and 2, we use the same neural structure as the one used for prediction, Eqs. (2.18–2.19). The network input is provided with the process inputs and outputs from the identification experiments and the model predictions. A normalized form of (2.20) over the prediction horizon is used as a cost function. It is denotes as the Relative Mean Square (RMS) error

$$RMS = \sqrt{\frac{\sum_{p=1}^{H_p} (d(k+p) - y_m(k+p/k))^2}{\sum_{p}^{H_p} (d(k+p))^2}},$$

where $y_m\ (k+p/k)$ denotes the predictions of the model output for the future sampling instant $k+p$ calculated at the current sampling instant k and $d(k+p)$is the process output collected during the identification experiment.

2.4 Identification Experiments

Data for NN training was collected according to the following experiments.

2.4.1 Classical (one test) Identification Experiment

Inputs (u_i) are generated as random signals. They are introduced to a dynamic crystallizer simulator used for laboratory experiments [24] and the respective process responses are recorded (y_i). The neural Models 1 and 2 are trained with data set $\{ u_i - u_{i,\mathrm{mean}}, y_i - y_{i,\mathrm{mean}}\}$, where ($u_{i,\mathrm{mean}}$, $y_{i,\mathrm{mean}}$) are the respective mean values of the collected input and output time series. This classical identification experiment is illustrated in Fig. 2.3 ($i = 1, 2$).

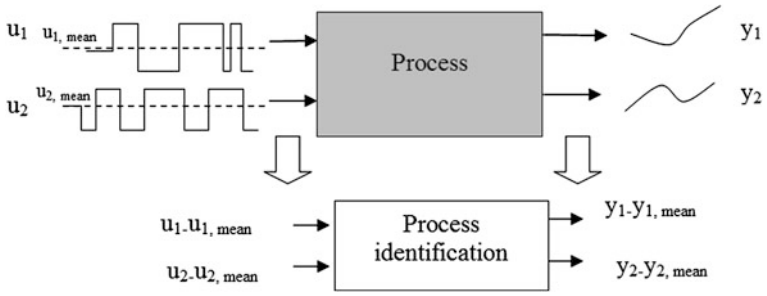

Fig. 2.3 Classical (one test) identification experiment

2.4.2 Double Test Identification

An alternative of the classical identification experiment was also studied. Prior to the test with randomly generated inputs (u_i), a test with constant inputs $\left(u_i' = u_{i,\text{mean}}\right)$ is performed and the process reactions $\left(y_i, y_i'\right)$ are recorded (Fig. 2.4). Then the neural models are trained with data set $\left\{u_i - u_i',\, y_i - y_i'\right\}$.

We expect that the second experiment has the potential to extract better the process dynamics.

Remark In case reference input trajectories are available (for example as a result of offline process design and optimization), the first test may be performed with these trajectories instead of constant inputs.

2.5 Error Tolerant MPC

2.5.1 Problem Formulation

In order to further relax the computational burden, a modification of the general NMPC problem (2.2), is proposed here with the following cost function

$$\left.\begin{aligned} &\min_{\Delta u(k/k),\Delta u(k+1/k),\ldots..\Delta u(k+H_c-1/k)} J = \\ &\lambda_1 \sum_{p=1}^{H_p} \left(e(k+p/k)\right)^2 + \lambda_2 \sum_{i=1}^{H_c} \left(\Delta u(k+p/k)\right)^2 \end{aligned}\right\}, \quad \text{if } E_\Sigma > \alpha,\ \alpha \in R^+ \tag{2.21}$$

$$\text{where } E_\Sigma = \frac{1}{H_p} \sum_{p=1}^{H_p} |e(k+p/k)| \tag{2.22}$$

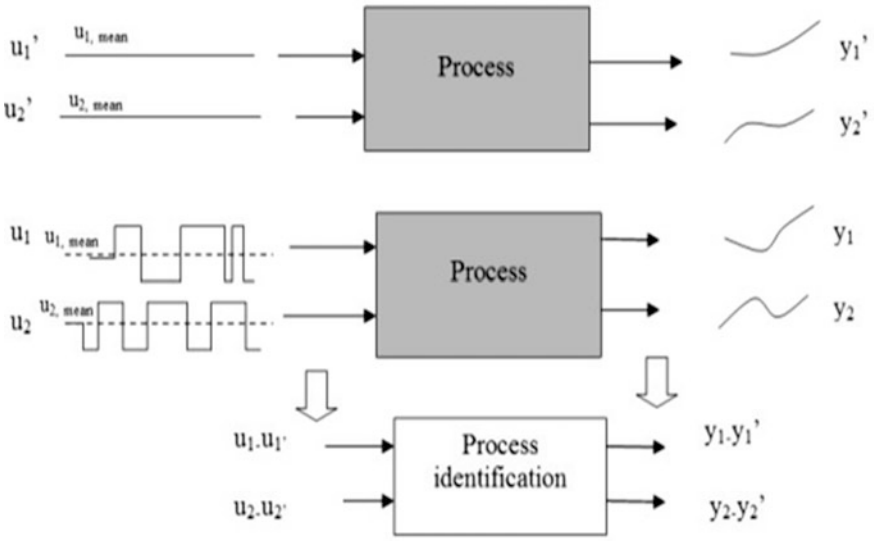

Fig. 2.4 Double-test identification experiment

subject to the same constraints (2.3–2.5). With the cost function (2.21) the future control actions are calculated as

$$u(k/k) = \begin{cases} u: \begin{cases} \min\limits_{[\Delta u(k/k),\Delta u(k+1/k),.....\Delta u(k+H_c-1/k))]} J = \\ \lambda_1 \sum\limits_{p=1}^{H_p} (e(k+p/k))^2 + \lambda_2 \sum\limits_{p=1}^{H_c} (\Delta u(k+p/k))^2, E_{\sum} > \alpha \end{cases} \\ u^* \qquad\qquad\qquad\qquad\qquad\qquad if\ E_{\sum} \leq \alpha \end{cases} \tag{2.23}$$

Equation (2.23) is a particular form of the general performance index defined by (2.2). We denote it as an error tolerant (ET) MPC because the optimization is performed only when the error function ($E_{\sum}$) is above a predefined real value α. When $E_{\sum}$ is inside the α-strip the control action is equal to u^*, the last implemented optimal control. The price to be paid is that the tracking is not achieved asymptotically, but in a neighborhood of the reference. However, the α tolerance can be arbitrarily small and is determined on a case by case basis, which suffices for practical purposes. The error function in (2.22) is defined as the mean of the errors between the predicted outputs and their reference values along the next H_p steps.

2.5.2 Selection of MPC Parameters

α is a design parameter whose choice is decisive for achieving a reasonable compromise between computational costs and tracking error. While a formal procedure for selecting is not defined, the error tolerance is chosen based on common sense consideration of $\pm 1 - \pm 5$ % error around the set-point.

The choice of H_p is related with the sampling period (Δt) of the digital control implementation, which in its turn is a function of the settling time t_s (the time before entering into the 5 % around the set-point) of the closed loop system. As a rule of thumb, it is suggested Δt to be chosen at least 10 times smaller than t_s, [18]. Hence, the prediction horizon can be chosen as $H_p = \mathbf{Int}\left(\frac{t_s}{\Delta t}\right)$. It is known that the smaller the sampling time, the better is the reference trajectory tracking and disturbance rejection. However, choosing a small sampling time yields a large prediction horizon. In order to compute the optimal control input, the optimization (2.2) is performed at each sampling time and requires large amount of computer memory per iteration and fast communication and computation resources. Such requirements are still unbearable for many industries with not-state-of-the-art control equipment. The ETMPC introduced by (2.23) has the potential to reduce the computational burden in such cases complementing other solutions in the literature for feasible real time optimal control.

In the MPC control tests (Sect. 2.8) the design parameter λ_1 is set to 1, while the choice of λ_2 is based on the following empirical expression

$$\lambda_2 = \frac{e_{\max} P/100}{(u_{\max} - u_{\min})^2} \tag{2.24}$$

where P defines the desired contribution of the second term in (2.21) (0 % $\leq$ P $\leq$ 100 %) and

$$e_{\max} = \max\left((ref - y_{\max})^2, (ref - y_{\min})^2\right) \tag{2.25}$$

The intuition behind (2.24–2.25) is to compensate the magnitude orders of the two terms of (2.21).

2.5.3 Normalized ETMPC

A normalized version of the cost function (2.21) was implemented in the numerical tests

$$\min_{\Delta u(k/k),\Delta u(k+1/k),.....\Delta u(k+H_c-1/k)} J = \lambda_1 RMS + \lambda_2 ACE$$
$$\textbf{if } E_{\Sigma} = \frac{1}{H_p} RMS > \alpha,\ \alpha \in R^{+} \tag{2.26}$$

where the Relative Mean Square (RMS) error is

$$RMS = \sqrt{\frac{\sum_{p=1}^{H_p} (ref(k+p/k) - \hat{y}(k+p/k))^2}{\sum_{i=1}^{H_p} (ref(k+p/k))^2}} \tag{2.27}$$

and the Average Control Effort (ACE)

$$ACE = \frac{\sum_{p=1}^{H_c} (\Delta u(k+p/k))^2}{H_c} \tag{2.28}$$

The normalization prevents from abrupt changes of the cost function and makes a balance between the two terms. The intuition behind is that RMS incorporates the constraints that the controlled output should not deviate too far from the reference values and ACE represent the assumption that the input variations are sufficiently slow. The cost function in (2.26) is smoother and guarantees empirically the Bounded Input Bounded Output (BIBO) stability of the closed loop system.

2.6 Sugar Crystallization Case Study

2.6.1 Process Description

Sugar production is characterized by strongly non-linear and non-stationary dynamics and goes naturally through a sequence of relatively independent stages: charging, concentration, seeding, setting the grain, crystallization, tightening and discharge [24].

The feedback control policy is based on measurements of the flowrate, the temperature, the pressure, the stirrer power and the supersaturation (by a refractometer). Measurements of these variables are usually available for a conventional crystallizer.

Charging. During the first stage the crystallizer is fed with liquor until it covers approximately 40 % of the vessel height. The process starts with vacuum pressure of around 1 bar (equal to the atmospheric pressure) and reduces it up to 0.23 bar.

When the vacuum pressure reaches 0.5 bar, the feed valve is completely open such that the feed flowrate is kept at its maximum value. When the liquor covers 40 % of the vessel height, the feed valve is closed and the vacuum pressure needs some time to stabilize around the value of 0.23 bar before the concentration stage starts.

Concentration. The next phase is the concentration. The liquor is concentrated by evaporation, under vacuum, until supersaturation reaches a predefined value (typically 1.11). At this stage seed crystals are introduced into the pan to induce the production of crystals. This is the beginning of the third (crystallisation) phase.

Crystallisation (main phase). In this phase as evaporation takes place further liquor or water is added to the pan in order to guarantee crystal growth at a controlled supersaturation level and to increase total contents of sugar in the pan. Near to the end of this phase and for economical reasons, the liquor is replaced by other juice of lower purity (termed syrup). The supersaturation is the main driving force of this stage but the actual measured variable is the brix of the solution. However, due to a straightforward relation between supersaturation and brix, the supersaturation (S) is considered as the controlled process output.

Tightening. Once the pan is full the feeding is closed. The tightening stage consists principally in waiting until the suspension reaches the reference consistency, which corresponds to a volume fraction of crystals equal to 0.5. The stage is over when the stirrer power reaches the maximum value of 50 A. The steam valve is closed, the water pump of the barometric condenser and the stirrer are turned off. Now the suspension is ready to be unloaded and centrifuged.

The different phases are comparatively independent process states. The crystallization is the main stage responsible for the product quality quantified by the average (in mass) crystal particle size (AM) at the end of the process and the final coefficient of particle variation (CV). The present study is focused on defining an efficient MPC only for the crystallization phase. Based on a set of industrial data collected over normal white sugar production cycles, average values for the main process variables were determined and summarised in Table 2.1. Table 2.2 consists of the reference values and restrictions of the process quality variables evaluated at the batch end. For more details with respect to the process see [25].

2.6.2 Crystallization Macro Model

The general phenomenological model of the fed-batch crystallization process consists of mass, energy and population balances. While the mass and energy balances are common expressions in many chemical process models, the population balance is related with the crystallization phenomenon which is still an open modelling problem.

Table 2.1 Process variables

Name	Notation	Average value	Max value
Liquor/Syrup feed flowrate	F_f	0.0057 m^3/s	0.025 m^3/s
Steam flowrate	F_s	1.6 m^3/s	2.75 kg/s
Feed temperature	T_f	65 °C	–
Steam temperature	T_s	140 °C	–
Brix of feed	Bx_f	0.7	–
Steam pressure	P_s	2 bar	–
Temperature of massecuite	T_m	72.5 °C	–
Vacuum pressure	P_{vac}	0.25 bar	0.5 bar
Brix of the solution	Bx_{sol}	2 bar	–
Stirring power	W	25 A	50 A

Table 2.2 Hard constraints of process quality variables

Name	Notation t_{final}–final time	Value
Average (in mass) crystal size	$AM(t_{final})$	0.55–0.6 mm (ref.)
Coef. of variation	$CV(t_{final})$	below 32 %
Volume	$V(t_{final})$	35 m^3 (max)
Supersaturation	S	1.3 (max)

2.6.2.1 Mass Balance:

The mass of all participating solid and dissolved substances are included in a set of conservation mass balance equations

$$\dot{M} = f(M(t), F(t), P1), \quad 0 \le t \le t_f, \quad M(0) = M_0 \tag{2.29}$$

where $M(t) \in R^q$ and $F(t) \in R^m$ are the mass and the flow rate vectors, with q and m dimensions respectively, and t_f is the final batch time. $P1$ is the vector of physical parameters as density, viscosity and purity.

2.6.2.2 Energy Balance:

The general energy (E) balance model is

$$\dot{E} = f(E(t), M(t), F(t), P2), \quad 0 \le t \le t_f, \quad E(0) = E_0 \tag{2.30}$$

where $P2$ incorporates the enthalpy terms and specific heat capacities derived as functions of physical and thermodynamic properties.

2.6.2.3 Population Balance:

Mathematical representation of the crystallization rate can be achieved through basic mass transfer considerations or by writing a population balance represented by its moment equations [26]. Employing a population balance is generally preferred since it allows to take into account initial experimental distributions and, most significantly, to consider complex mechanisms such as those of size dispersion and/or particle agglomeration/aggregation. Hence

$$\dot{\eta}_i = f\left(\eta_i(t), \tilde{B}_0, G, \beta^{'}\right), \quad 0 \le t \le t_f, \quad i = 0, 1, 2, \ldots \eta_i(0) = \eta_{i0} \tag{2.31}$$

where η_i is the j-th moment of the mass-size particle distribution function, $\tilde{B}_0$, G and β' are the kinetic variables nucleation rate, linear growth rate and the agglomeration kernel, respectively. The process quality measures are derived as

$$AM(t) = \eta_1(t)/\eta_0(t) \tag{2.32}$$

$$CV(t) = \left(\eta_0(t)\eta_2(t)/\eta_1^2(t) - 1\right)^{1/2} \tag{2.33}$$

The control objective is to get a desired final average crystal size $AM(t_{final}) = AM_{ref} = 0.55$ and to minimize $CV(t_{final})$.

2.7 Neural Model Identification

2.7.1 Identification Experiments

Based on the macro model (2.29–2.33), a dynamical process simulator was developed [24]. It consists of mass, energy and population balances and was extensively tuned with real data from two industrial units (Sugar refinery RAR.SA, Portugal and Company 30 de Noviembre, Pinar del Río, Cuba). For all measured process variables (the vacuum pressure, brix and temperature of the feed flow, pressure and temperature of the steam), the simulator provides an option to introduce industrial measurements, which serves as a natural source of disturbance and noise. The results of the MPC tests are related with the Sugar refinery RAR.SA plant.

Over the crystallization stage the controlled variable is the supersaturation (S). The manipulated variables are the feed flow rate (F_f), during the first part of the crystallization and latter on the steam flow rate (F_s). The open-loop input-output process data were acquired either by available industrial measurements (batch 1, 2, 3) or by generated inputs as random signals limited by maximum of 10 % around the average values collected in Table 2.1. In both cases the sampling period is $t_s = 10$ s.

Data bases for NN training and testing were obtained according to the single or double test experiments discussed in Sect. 2.4.

2.7.2 NN Structure Selection

In term of NN structure selection, the number of delayed signals (n_y and n_u) used as regressors (lag space) and the number and type of hidden units need to be determined.

2.7.2.1 Lag Space Selection

Insufficient lag space means that essential dynamics is not modeled. Too many regressors imply overestimated model order. A procedure based on the Lipschitz quotient is used to optimize the lag space, assuming that the process can be represented accurately by a function that is reasonable smooth in the regressors. A detailed description of this approach can be found in [13] and [27]. The procedure is the following:

1. For a given choice of the lag space n_y determine the Lipschitz quotients for all combinations of input-output pairs

$$q_{ij} = \left| \frac{y_i^{NN}(k) - y_j^{NN}(k)}{x_i(k) - x_j(k)} \right|, \quad i \neq j, \quad i,j = 1,2,\ldots..N \tag{2.34}$$

where || specifies the Euclidian norm, i.e. the distance, N is the number of available samples, $x_i(k)$ is one element of the input vector $x(k)$

$$y^{NN}(k) = g[x(k), \theta], \quad x(k) = [x_1, x_2, \ldots..x_z] \tag{2.35}$$

The Lipschitz condition states that q_{ij} is always bounded if the function g is continuous.

2. Select the p largest quotients, where $p = 0.01\ N \div 0.02\ N$
3. Evaluate the criterion

$$q = \left(\prod_{i=1}^{p} \sqrt{n_y} q_i^{(n_y)} \right)^{\frac{1}{p}} \tag{2.36}$$

4. Repeat (1–3) for a number of different lag structures.
5. Plot the criterion q as a function of the lag space and select the optimal number of regressors as the "knee point" of the curve.

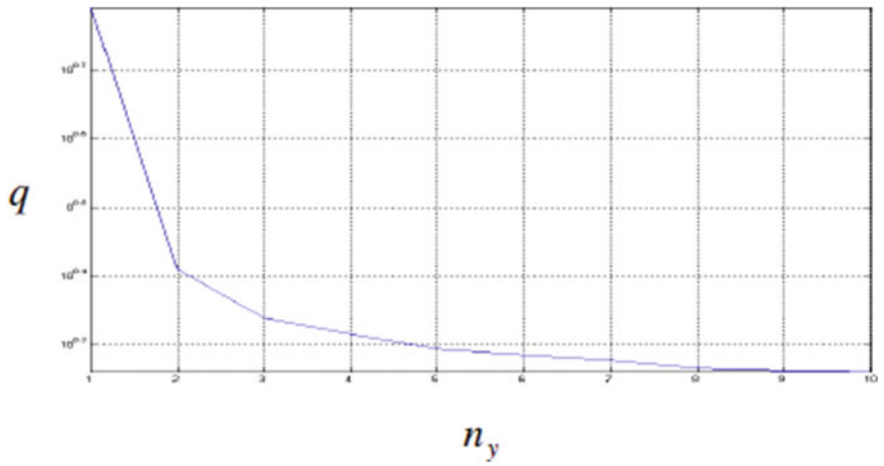

Fig. 2.5 Criterion (2.36) as a function of the lag space n_y

For the crystallization case

$$x(k) = \left[y(k-1), \ldots y(k-n_y), u(k-\tau), \ldots u(k-\tau-n_u+1)\right] \tag{2.37}$$

The process is simplified by choosing $n_y = n_u$. The crystallization stage takes about 45–50 min., with a sampling rate $t_s = 10$ s., hence $N \approx 300$ samples and $p = 3 \div 6$. Based on the results depicted in Fig. 2.5, third order dynamical model was defined.

2.7.2.2 Number of Hidden Nodes

The studied neural models have equal input arguments $n_y = n_u = 3$, $\tau = 1$, and hyperbolic tangent neurons in the hidden layer. Table 2.3 summarized the results with respect to neural model training and testing as a function of the number of hidden nodes (H). Classical training algorithms have been tested: steepest descent (SD), conjugate gradient (CG) [28] and Levenberg-Marquardt (LM) method [29]. In terms of RMS, the methods are similar, however the LM method converges most rapidly. Hence, the LM training suits better for on-line neural model tuning or periodical on-line model calibration.

More hidden nodes H (3, 4, 5, 6, 7), lower RMS for the training data, however for H > 6, the RMS for test data rapidly increases (overfitting problem). Therefore H = 6 is chosen as a compromise between accuracy and complexity.

Table 2.3 The effect of the number of hidden nodes and the training algorithm on the accuracy of the neural dynamical model

H	No. of weights	Training method	*RMS* training	*RMS* test
3	19	SD	0.195	0.289
		CG	0.189	0.296
		LM	0.201	0.311
4	25	SD	0.087	0.134
		CG	0.141	0.199
		LM	0.109	0.220
5	31	SD	0.079	0.097
		CG	0.056	0.261
		LM	0.191	0.301
6	37	SD	0.068	0.091
		CG	0.011	0.101
		LM	0.026	0.099
7	43	SD	0.066	0.172
		CG	0.009	0.391
		LM	0.017	0.283

2.7.2.3 NN Pruning

Besides the number of regressors and the number of hidden nodes, the number of connections among the neurons is also an important NN parameter. The Optimal Brain Damage (OBD) pruning algorithm [30] is used to further improve the ratio between accuracy and complexity

OBD determines the optimal network architecture by removing the superfluous weights from the network in order to avoid the overfitting of the data by the NN. The objective is to find a set of weights whose removing is likely to result in the least change in RMS. The saliency S_i of weight ϑ_i is defined as a function of the second-order derivative of the cost function with respect to $\vartheta_i \left(\frac{\partial^2 RMS}{\partial(\theta_i)^2}\right)$

$$S_i = \frac{\partial^2 RMS}{\partial(\theta_i)^2}(\theta_i)^2, \quad i = \{1, 2, \ldots n\} \quad n = (n_y + n_u + 1)H + H + 1 \tag{2.38}$$

where n is the total number of network weights and

$$\vartheta = [\theta_1, \theta_2, \ldots \theta_n] = \left[\{a_j\}, \{b_{ji}\}, \{c_{jk}\}\right],$$
$$\text{for} \quad j = 0, \ldots H, \; i = 0, \ldots n_y, k = 1, \ldots n_u$$

Table 2.4 The effect of NN prunning on the final process quality measures

Method	No weights	*AM_final* (mm) (ref. 0.56)	*CV_final* (%)
NN MPC	37 (before pruning)	0.615	30.28
NN ETMPC		0.603	31.14
Normalized NN ETMPC		0.573	29.36
NN MPC	23 (after pruning)	0.637	30.26
NN ETMPC		0.636	30.42
Normalized NN ETMPC		0.550	28.74

Table 2.5 MPC design parameters

t_s (s) Settling time	Δt (s) Sampling period	$\boldsymbol{H}_p$ Prediction horizon	$\boldsymbol{H}_c$ Control horizon	λ_2 Weight	Set-point	α-strip (1 %)
Control variable-feed flowrate $\boldsymbol{F}_f$; controlled variable-supersaturation S						
40	4	10	4	0.1	1.15	0.01
PID: proportional gain $k_p = -0.5$; integral time const $\tau_i = 40$; $\tau_d = 0$						
Control variable-feed flowrate $\boldsymbol{F}_s$; controlled variable-supersaturation S						
60	4	15	4	0.01	1.15	0.01
PID: proportional gain $k_p = 20$; integral time const $\tau_i = 40$; $\tau_d = 0$						

The OBD pruning algorithm can be summarized as follows:

1. Train the fully connected NN
2. Compute the saliency (2.38) with respect to all weights
3. Eliminate the weight with the smallest saliency value
4. Retrain the NN
5. Stop if pruning leads to higher RMS
6. If not, go to step 2.

The NN started with 6 hidden nodes and 37 weights. After the OBD pruning procedure, 14 weights were removed but the network predictions of the end-point process quality measures *AM_final* and *CV_final* are still reliable (see Table 2.4).

2.8 NN ETMPC Control Tests

The NN-based MPC was tested on a dynamic crystallizer simulator [24]. The MPC design parameters are summarized in Table 2.5. The NN MPC with the classical cost function (2.2) and the Error Tolerant modifications (not normalized and normalized NN ETMPC) with cost functions (2.21) and (2.26) respectively, are compared with a PID controller designed in the following velocity form

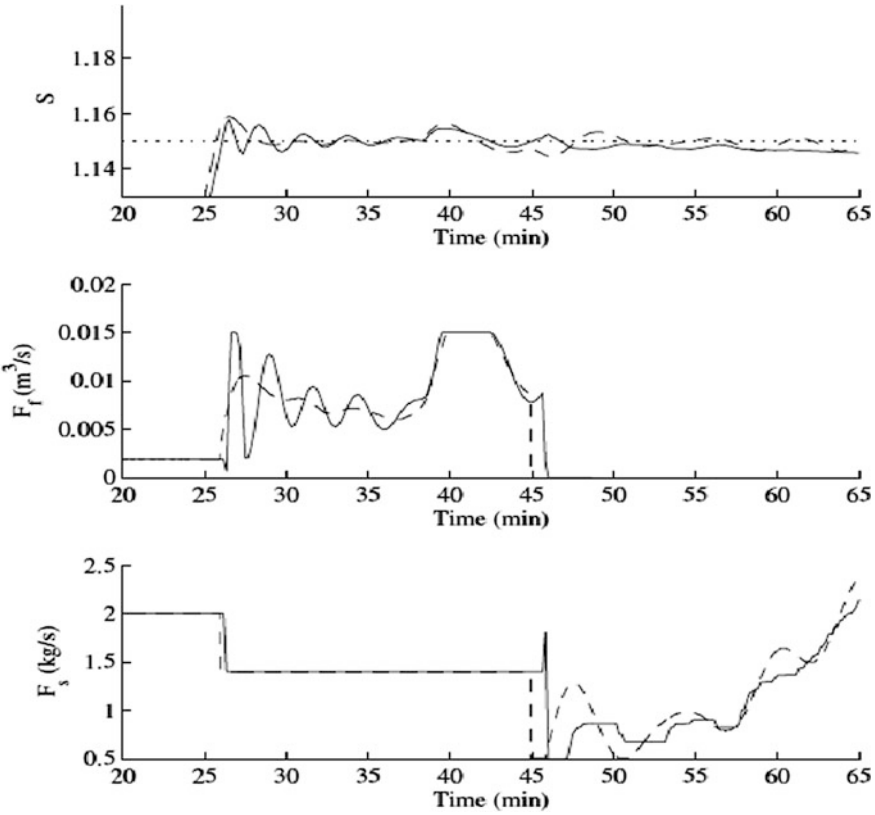

Fig. 2.6 Controlled variable (S-supersaturation), manipulated variables (Ff-feed flowrate), Fs-steam flow rate. *Dashed line*—NN-Normalized ETMPC; *Full line*—NN-MPC

$$u(k) = u(k-1) + K_p\Bigg((e(k) - e(k-1)) + \frac{\Delta t}{\tau_i}\cdot e(k) + \frac{\tau_d}{\Delta t}\cdot (e(k) - 2\cdot e(k-1) + e(k-2))\Bigg) \tag{2.39}$$

The optimized PID parameters k_p, τ_i, τ_d are summarized in Table 2.5. The tuning suggests that the derivative time constant is not necessary therefore $\tau_d = 0$.

2.8.1 Set-Point Tracking

Time trajectories of the controlled and the manipulated variables only for the crystallization stage are depicted in Figs. 2.6, 2.7, 2.8. During the first half of the stage liquor is introduced into the pan and its flow rate (F_f) is the way to control

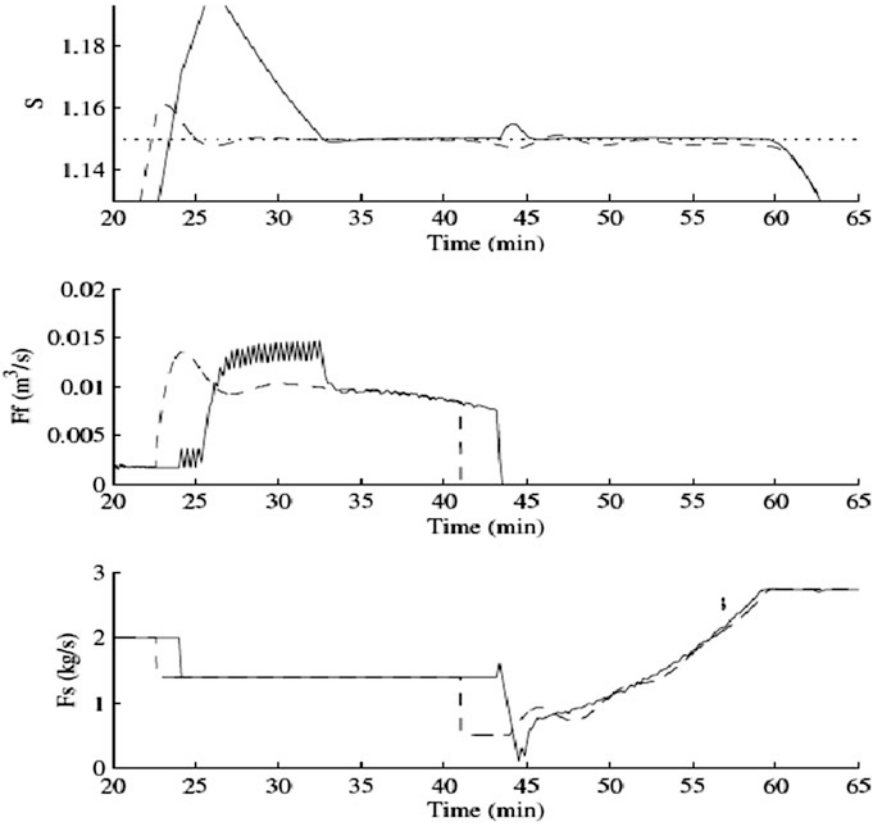

Fig. 2.7 Controlled variable (S-supersaturation), manipulated variables (Ff-feed flowrate), Fs-steam flow rate. *Dashed line*—NN-Normalized ETMPC; *Full line*—NN-ETMPC

the supersaturation around the set point. When the complete amount of liqueur is added, the process is next controlled by the steam flow rate (F_s).

All MPC control loops show similar tracking performance. There results demonstrate that the explicitly introduced error tolerance does not worsen the output reference tracking. The price to be paid is that the (not normalized) NN ETMPC needs longer time to enter into the α-strip, however, it is not the case with the normalized NN ETMPC. The PID exhibits the worse tracking performance (Fig. 2.8—full line) which is due to the inherent nonstationarity of the process, while the PID parameters are tuned assuming stationary process parameters.

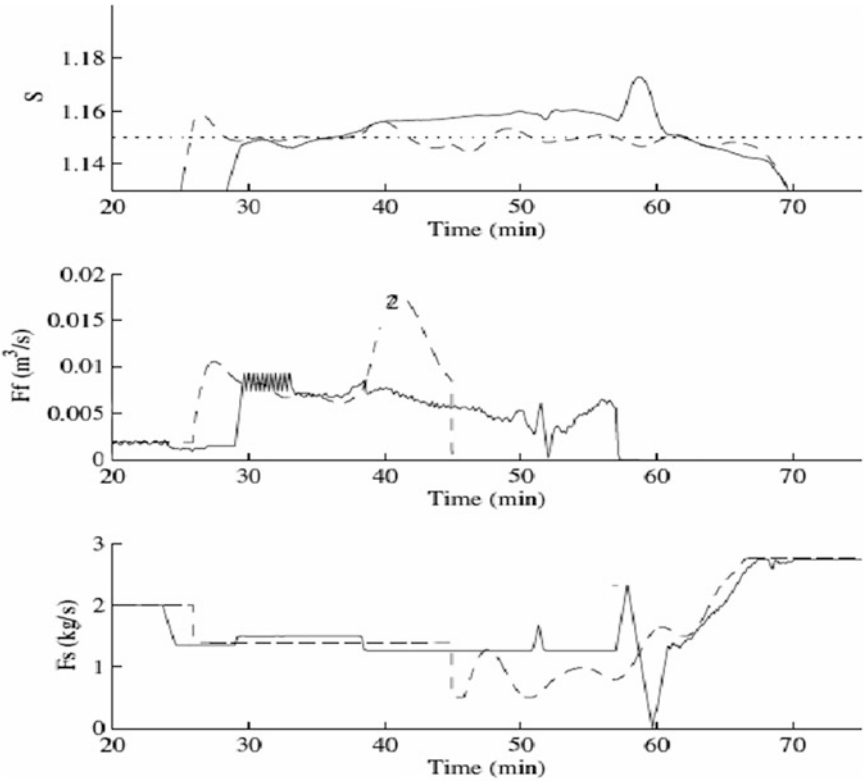

Fig. 2.8 Controlled variable (S-supersaturation), manipulated variables (Ff-feed flowrate), (Fs-steam flow rate). *Dashed line*—first principles model-Normalized ETMPC; *Full line*—PID

2.8.2 Computational Time Reduction

The principal contribution of the NN-based Error Tolerant MPC is with respect to the computational time reduction. Figures 2.9 and 2.10 depict the optimization time per iteration over one production cycle, comparing the classical MPC (Fig. 2.9a) with tree ETMPC alternatives: i) ETMPC with neural Model 1, Eq. (2.18); ii) ETMPC with neural Model 2, Eq. (2.19); iii) Normalized ETMPC with neural Model 1. The NN-based ETMPCs require significantly less time to compute the feasible (sub) optimal control actions. The average CPU times per iteration are summarized in Table 2.6. Note that they range from about 0.211–0.192 s for the classical MPC to 0.061–0.078 s for the Normalized NN-ETMPC, assuming only 1 % error tolerance. In case the process tolerates higher tracking imprecision the computational time can be further reduced.

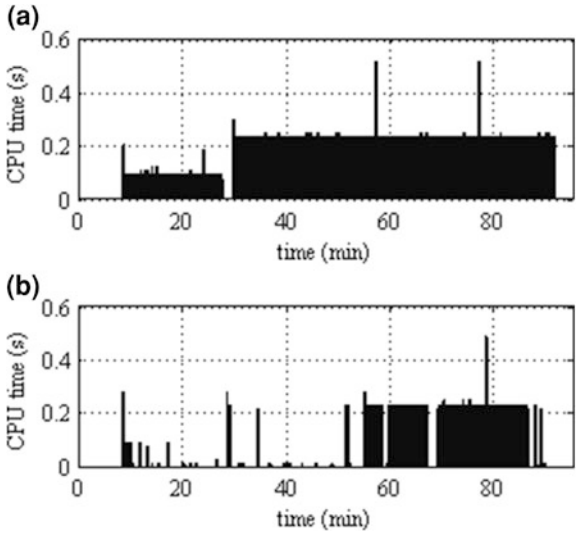

Fig. 2.9 Optimization time per iteration. **a** NN-MPC **b** NN (Model 1)-ETMPC

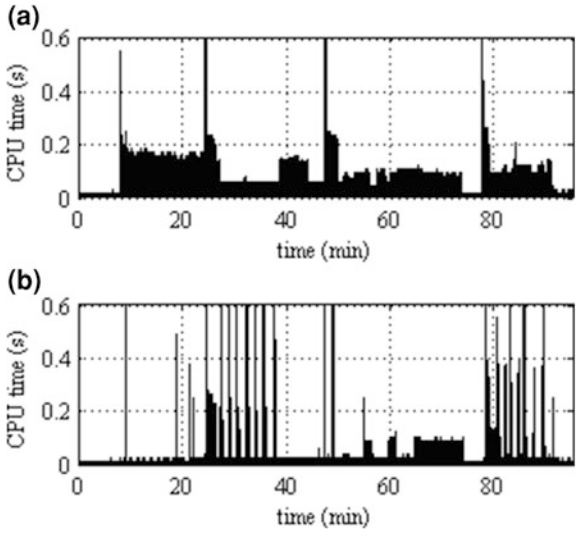

Fig. 2.10 Optimization time per iteration. **a** NN (Model 2)-ETMPC; **b** Normalized NN (Model 1)-ETMPC

2.8.3 Final Product Quality

For practical applications it is interesting to study the influence of the NN predictive models on the process final performance. Therefore, their prediction quality is tested in the framework of the normalized NN ETMPC towards the principal control objectives, namely how close is the average crystal size (AM_final) to the desired value of 0.55 mm. and how variable is this size (CV_final) at the end of the process. Tables 2.7 and 2.8 collect the performed tests. First (Table 2.7), Models 1

Table 2.6 Average computational cost

Control method	NN Model	Average optim. time per iteration (S)
Normalized NN ETMPC	Model 1/eq. (2.18)	0.061
	Model 2/eq. (2.19)	0.078
NN ETMPC	Model 1	0.093
	Model 2	0.091
NN MPC	Model 1	0.192
	Model 2	0.211

Table 2.7 NN prediction quality in the framework of the normalized ETMPC

Model	Training data base	AM_final (mm) (ref. 0.55)	CV_final (%)
Model 1	1 batch (273 p.)	0.615	31.14
	2 batches (551 p.)	0.592	30.93
	3 batches (816 p.)	0.550	28.74
Model 2	1 batch (373 p.)	0.61	29.18
	2 batches (551 p.)	0.575	32.06
	3 batches (816 p.)	0.601	30.10

Table 2.8 Process model identification assuming model 1 structure

Identif.	Training data base	RMS test	AM_fin (mm) (ref. 0.55)	CV_final (%)
One test	Generated signals	0.256	0.644	30.19
	Industrial measurements	0.269	0.587	32.01
Double test	Generated signals	0.097	0.591	29.06
	Industrial measurements	0.102	0.550	28.74

and 2 were trained with industrial data acquired over one, two or three in spec batches. Increasing the training data influences the prediction quality mainly of Model 1 whose structure depends directly on the process measurements. In case less training data are available Model 2 exhibits better prediction ability. The effect of the identification strategies discussed in Sect. 2.4 is studied assuming an MPC controlled process with Model 1 predictions (Table 2.8). The double test identification is less sensitive with respect to the nature of the training data (generated signals or industrial measurements) and ends up with lower RMS.

2.9 Conclusions

The traditional position of many practitioners is still in favor of linear control solutions. However, for significantly nonlinear processes, as for example in the crystallization industry, the advantages of data-based process modeling,

optimization and control are difficult to neglect. This chapter discusses the application of NNs as predictive models in the framework of nonlinear model based predictive control (MPC).

The proposed recurrent multistep ahead predictive neural model fits well to the MPC objectives and guarantees the same performance as the traditional models obtained from physical laws. Furthermore, the multi step ahead prediction role of the model in the MPC framework is taken into account during the NN training. These modifications, of the classical regression predictive models, reduce the propagation of the prediction error and have the advantage to be a straightforward modeling technology.

The second contribution of the present work is the heuristic modification of the MPC, termed ETMPC, where the optimization is executed only when the tracking error is above a predefined level. In applications where the tracking is allowed to tolerate some margins, the suboptimal NN-based ETMPC control can relax substantially the computational burden. In the presence of limited computational resources suboptimal control is often the compromise due to feasibility problems or lack of solution (convergence) within the restricted time per iteration.

From a systems engineering point of view, the crystallization phenomena can be found in a significant number of industrial processes (e.g. pharmaceutical industry, precipitation, wastewater treatment). Therefore, the control solution proposed for the present industrial case has the potential to be easily extended to other processes.

Acknowledgements The work was partially funded by the Portuguese National Foundation for Science and Technology (FCT) in the context of the project FCOMP-01-0124-FEDER-022682 (FCT reference PEst-C/EEI/UI0127/2011) and the Institute of Electrical Engineering and Telematics of Aveiro (IEETA), Portugal.

References

1. Clarke, D.W., Mohtadi, C., Tuffs, P.S.: Generalized predictive control. part 1. The basic algorithm. part 2: Extensions and interpretations. Automatica **23**, 137–148 (1987)
2. Rossiter J.A.: Model-based predictive control: a practical approach. CRC press, Boca Raton. (2003) ISBN 0-9493-1291-4
3. Morari, M.: Advances in model-based predictive control. Oxford University Press, Oxford (1994)
4. Balasubramhanya, L.S., Doyle III, F.J.: Nonlinear model-based control of a batch reactive distillation column, J. Process Control. **10,** 209–218 (2000)
5. Nagy, Z.K., Braatz, R.D.: Robust nonlinear model predictive control of batch processes. AIChE J. **49**, 1776–1786 (2003)
6. Nagy, Z.K.: Model based control of a yeast fermentation bioreactors using optimally designed artificial neural networks. Chem. Eng. J **127**(1), 95–109 (2007)
7. Qin, S.J., Badgewell, T.: A survey of industrial model predictive control technology. Control Eng. Pract **11**, 733–764 (2003)
8. Hunt, K.J., Sbarbaro, D., Zbikowski, R., Gawthrop, P.J.: Neural networks for control systems-a survey. Automatica **28**, 1083–1112 (1992)

9. Nørgaard, M.: System identification and control with neural networks. PhD thesis, Department of Automation, Technical University of Denmark, Lyngby (1996)
10. ŁAwryńCzuk, M.: A family of model predictive control algorithms with artificial neural networks. Int J. Appl Math Comput. Sci. **17**(2), 217–232 (2007)
11. ŁAwryńczuk, M., Tatjewskin, P.: Nonlinear predictive control based on neural multi models, Int. J. Appl. Math. Comput. Sci. **20**(1), 7–21 (2010)
12. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Networks **1**(1), 4–27 (1990)
13. Nørgaard, M., Ravn, O., Poulsen, N.K., Hansen, L.K.: Neural networks for modelling and control of dynamic systems. Springer, London (2000)
14. Morari, M., Lee, J.H.: Model predictive control: Past, present and future. In: Proceedings of the PSE'97-ESCAPE-7 Symposium, Trondheim (1997)
15. Nagy, Z.K., Mahn, B., Franke, R., Allgöwer, F.: Nonlinear model predictive control of batch processes: An industrial case study, IFAC World Congress, Prague (2005)
16. Lightbody, G., Irwin, G.W.: Nonlinear control structures based on embedded neural system models. IEEE Trans. Neural Networks **8**(3), 553–567 (1997)
17. Hunt, K.J. Sbarbaro, D.: Neural networks for nonlinear internal model control".IEE Proceedings-D **138**(5), 431–438 (1991)
18. Georgieva, P., Feyo de Azevedo, S.: Novel computational methods for modeling and control in chemical and biochemical process systems. In: do Carmo Nicoletti, M., Jain, L.C. (eds.) Comp. Intelligence Techniques for Bioprocess Modelling, Supervison and Control, SCI 218, Springer, Heidelberg, 99–125 (2009)
19. Parisini, T., Sanguineti, M., Zoppoli, R.: Nonlinear stabilization by receding-horizon neural regulators. Int. J. Control **70**(3), 341–362 (1998)
20. Lu, C.H., Tsai, C.C.: Adaptive predictive control with recurrent neural network for industrial processes: An application to temperature control of a variable-frequency oil-cooling machine. IEEE Trans. Industr. Electron. **55**(3), 1366–1375 (2008)
21. Peng, H., Yang, Z.J., Gui, W., Wu, M., Shioya, H., Nakano, K.+ : Nonlinear system modeling and robust predictive control based on RBF-ARX model. Eng. Appl.Artif. Intell. **20**(1), 1–9 (2007)
22. Hagan, M.T., Demuth, H.B., Beale, M.H.: Neural Network Design. PWS Publishing Company, Pacific Grove (1996)
23. Mandic, D.P., Chambers, J.A.: Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability (Adaptive & Learning Systems for Signal Processing, Communications & Control), Wiley, New York (2001)
24. Georgieva, P., Meireles, M.J., Feyo de Azevedo, S.: Knowledge-based hybrid modelling of batch crystallization when accounting for nucleation, growth and agglorameration phenomena. Chem. Eng. Sci. **58**, 3699–3713 (2003)
25. Oliveira, C., Georgieva, P., Rocha, F., Feyo de Azevedo, S.: Artificial neural networks for modeling in reaction process systems. Neural Comput. Appl. **18**, 15–24 (2009)
26. Paz Suárez, L.A., Georgieva, P., Feyo de Azevedo, S.: Nonlinear MPC for fed-batch multiple stages sugar crystallization, IChemE. Chem. Eng. Res. Des. (2010). doi:10.1016/j.cherd.2010.10.010
27. He, X., Asada, H.: A new method for identifying orders of input-output models for nonlinear dynamic systems, In: Proceedings of the American Control Conference, San Diego (1993)
28. Zhou, G., Si, J.: Advanced neural-network training algorithm with reduced complexity based on jacobian deficiency. IEEE Trans. Neural Networks **9**(3), 448–453 (1998)
29. Lera, G., Pinzolas, M.: Neighborhood based levenberg-marquardt algorithm for neural network training. IEEE Trans. Neural Networks **13**(5), 1200–1203 (2002)
30. Reed, R.: Pruning algorithms—a survey. IEEE Trans. Neural Networks **4**(5), 740–747 (1993)

http://www.springer.com/978-3-662-43369-0

Innovations in Intelligent Machines-5
Computational Intelligence in Control Systems Engineering
Balas, V.E.; Koprinkova-Hristova, P.; Jain, L.C. (Eds.)
2014, XV, 248 p. 129 illus., Hardcover
ISBN: 978-3-662-43369-0